Geometric Coherence in Context-Addressed Linear Mapping Networks

Abstract

Viewing neural networks as **context-addressed linear mapping systems** exposes a much larger architectural design space than is usually considered. This viewpoint also suggests an important design principle: preserving **geometric coherence** is often more important than maximizing switching flexibility.

Topology Is Not Enough

Continuity and topology alone do not explain why certain architectures train well.

A dense random linear mapping is perfectly continuous, yet it destroys important geometric structure.

For example:

- Small image translations become complicated global changes after a dense random projection.
- Local spatial relationships become distributed across every output dimension.
- Optimization must first reconstruct useful geometry before useful learning can occur.

The issue is not continuity but preservation of the natural transformations of the problem.

Geometric Coherence

A useful family of linear mappings should satisfy:

- Nearby inputs produce nearby computations.
- Nearby contexts produce nearby linear mappings.
- Natural transformations (translation, rotation, temporal evolution, etc.) remain simple after transformation.

The computational geometry should evolve smoothly across both input space and context space.

What ReLU Networks Teach Us

ReLU networks are Binary Context Addressed Linear Mapping (BCALM) systems.

However, their switching mechanism is remarkably restrictive.

Each ReLU bit simply masks one output component. Consequently:

- entire rows of the effective linear operator are either enabled or disabled,
- individual matrix elements cannot be switched independently,
- many linear mappings differ only by row removal.

From a computational viewpoint, this is a surprisingly crude addressing mechanism.

Despite this limitation, ReLU networks perform extremely well.

The obvious conclusion is that **backpropagation is highly effective at organizing families of piecewise-linear mappings into coherent computational systems**, even when the context-addressing mechanism itself is far from optimal.

This suggests that while increasingly complex context selectors are possible and should be explored, the bias should be toward simplicity if optimization can organize the resulting family of mappings.

A Larger Design Space

Once context selection is separated from the choice of linear operator, many architectures become possible.

The selected operator could be:

- dense matrices,
- convolution operators,
- sparse operators,
- wavelet transforms,
- low-rank operators,
- block-structured operators,
- graph operators.

Likewise, the context may come from:

- learned binary features,
- robot state,
- time,
- external sensors,
- finite-state machines,
- locality-sensitive projections.

These design choices are largely independent.

Switched Structured Operators

One particularly attractive direction is **context-switched structured operators**.

Rather than selecting arbitrary dense matrices, context may select among structured operators such as convolution kernels.

Advantages include:

- preservation of translation geometry,
- context specialization without abandoning locality,
- reduced burden on optimization to reconstruct useful geometry.

More generally, the selected linear mappings should possess useful geometric structure before learning begins.

Design Guidelines

The observations suggest several practical principles.

Prefer architectures that:

- preserve natural geometric transformations,
- maintain locality where appropriate,
- vary computation gradually with context,
- exploit structured linear operators.

Avoid architectures that:

- randomize geometry unnecessarily,
- introduce abrupt computational changes for small context changes,
- maximize flexibility while discarding useful structure.

Hypothesis

Backpropagation is not merely an optimizer of parameters; it is an organizer of computational geometry.

The role of the architecture is therefore to provide a family of candidate linear mappings whose geometry is compatible with efficient optimization, rather than attempting to maximize the number of independently switchable parameters.